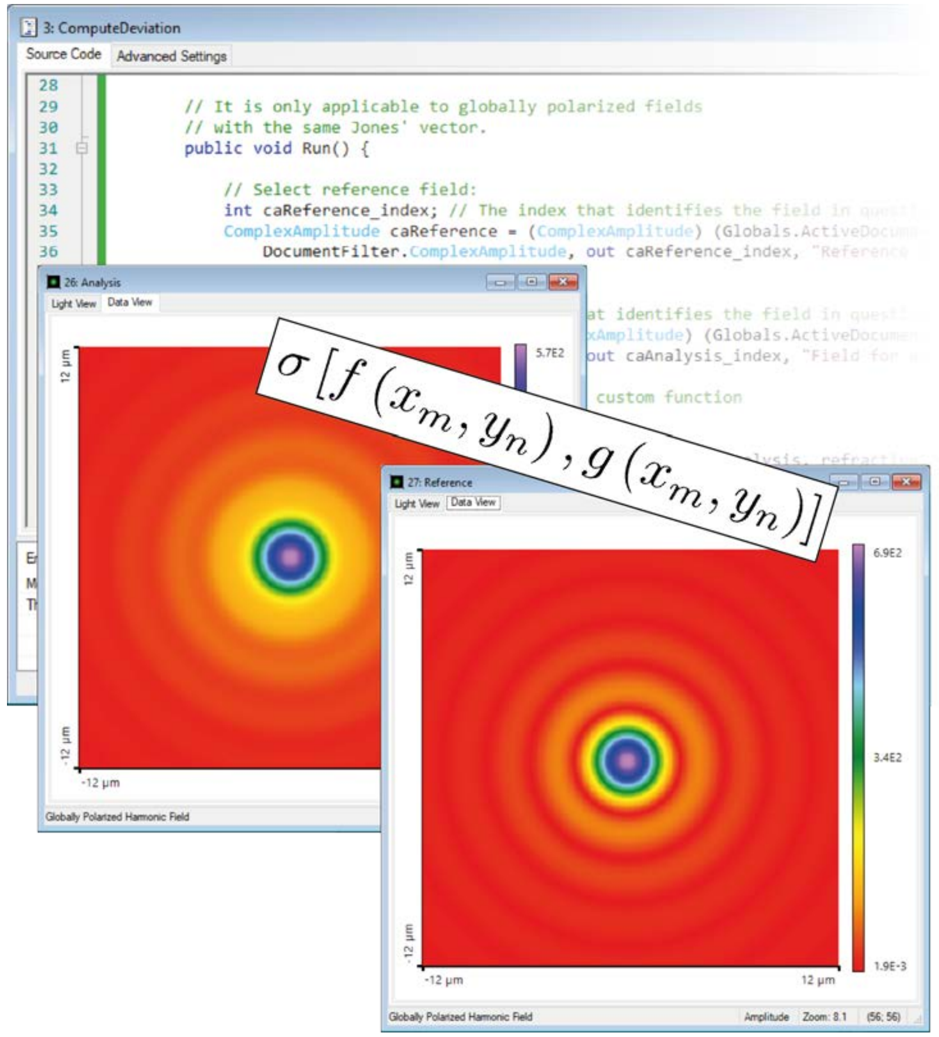
Programming a Module That Computes the Standard Deviation between Two Harmonic Fields

Abstract



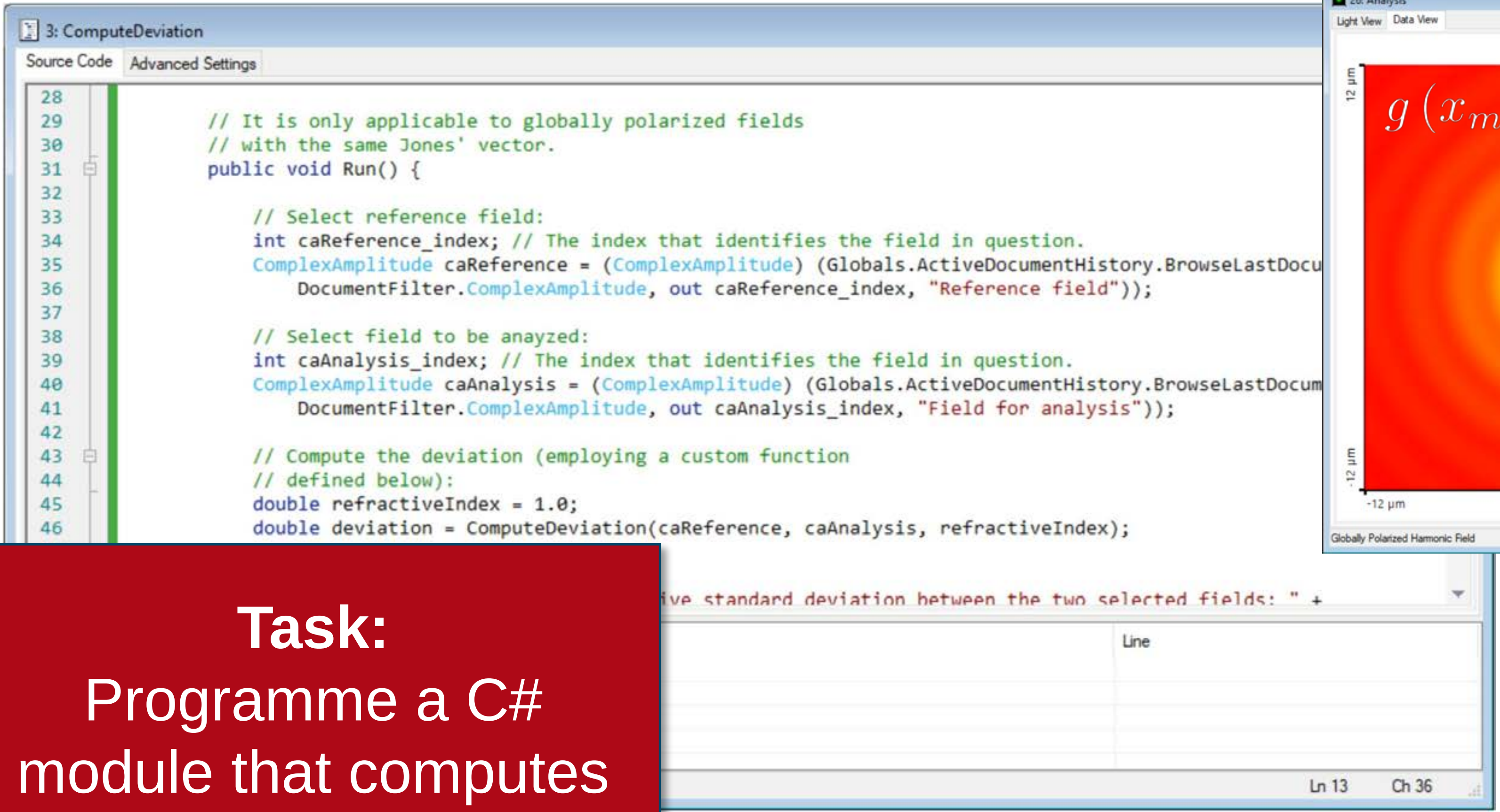
```
        // It is only applicable to globally polarized fields
        // with the same Jones' vector.
        public void Run() {

            // Select reference field:
            int caReference_index; // The index that identifies the field in questi
            ComplexAmplitude caReference = (ComplexAmplitude) (Globals.ActiveDocum
                DocumentFilter.ComplexAmplitude, out caReference_index, "Reference
```

$$\sigma\left[f\left(x_m, y_n\right), g\left(x_m, y_n\right)\right]$$

Being able to gauge the accuracy of a given result with respect to another which is taken as the reference is a fundamental feature in science and engineering. In this use case, we show an example in VirtualLab Fusion of a custom module that allows the user to compute the standard deviation of a field mode with respect to another one. The module permits the user to select both fields from the open documents in the session and yields the result in the Messages tab.

Programming a Module that Computes the Standard Deviation



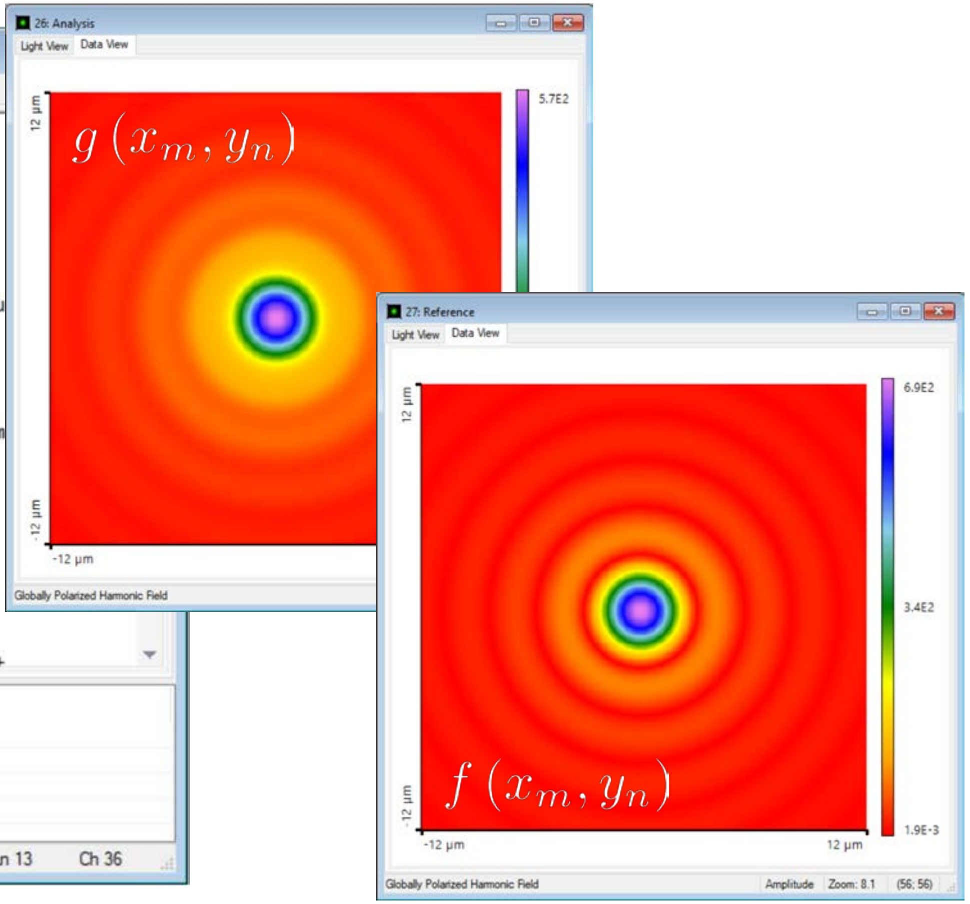
```
        // It is only applicable to globally polarized fields
        // with the same Jones' vector.
        public void Run() {

            // Select reference field:
            int caReference_index; // The index that identifies the field in question.
            ComplexAmplitude caReference = (ComplexAmplitude) (Globals.ActiveDocumentHistory.BrowseLastDocu
                DocumentFilter.ComplexAmplitude, out caReference_index, "Reference field"));

            // Select field to be anayzed:
            int caAnalysis_index; // The index that identifies the field in question.
            ComplexAmplitude caAnalysis = (ComplexAmplitude) (Globals.ActiveDocumentHistory.BrowseLastDocum
                DocumentFilter.ComplexAmplitude, out caAnalysis_index, "Field for analysis"));

            // Compute the deviation (employing a custom function
            // defined below):
            double refractiveIndex = 1.0;
            double deviation = ComputeDeviation(caReference, caAnalysis, refractiveIndex);

ive standard deviation between the two selected fields: " +
```



Task:
Programme a C# module that computes the deviation between two fields.

$$\sigma\left[f\left(x_m, y_n\right), g\left(x_m, y_n\right)\right] = \frac{\sum\left|f\left(x_m, y_n\right) - g\left(x_m, y_n\right)\right|^2 \delta x \delta y}{\sum\left|f\left(x_m, y_n\right)\right|^2 \delta x \delta y}$$

Programming a Module that Computes the Standard Deviation

C# Module: Header

```
using System;
// Keep all default usings.
using VirtualLabAPI.Core.Propagation;

namespace OwnCode {
    public class VLModule : IVLModule {

        // This module computes the standard deviation between two complex amplitude fields. The user can select both of them,
        // first the reference, then the one to be compared against the reference.

        // It is only applicable to globally polarized fields with the same Jones' vector.

        // Please note that there exists an in-built function in VirtualLab Fusion for the calculation of the standard deviation,
        // ComplexAmplitudeOperations.ComputeDeviation. This module serves as a programming example.
```

Programming a Module that Computes the Standard Deviation

C# Module: Main Function

```
public void Run()
        {

            // Select reference field:
            int caReference_index; // The index that identifies the field in question.
            ComplexAmplitude caReference = (ComplexAmplitude) (Globals.ActiveDocumentHistory.BrowseLastDocuments(
                DocumentFilter.ComplexAmplitude, out caReference_index, "Reference field"));

            // Select field to be anayzed:
            int caAnalysis_index; // The index that identifies the field in question.
            ComplexAmplitude caAnalysis = (ComplexAmplitude) (Globals.ActiveDocumentHistory.BrowseLastDocuments(
                DocumentFilter.ComplexAmplitude, out caAnalysis_index, "Field for analysis"));

            // Compute the deviation (employing a custom function
            // defined below):
            double refractiveIndex = 1.0;
            double deviation = ComputeDeviation(caReference, caAnalysis, refractiveIndex);

            // Display the result:
            VL_GUI.WriteLineToMessagesTab("Relative standard deviation between the two selected fields: " +
                deviation.ToString());
        }
```

Programming a Module that Computes the Standard Deviation

C# Module: Support Function (Part I)

```
        /// <summary>
        /// Function to compute the standard deviation between two
        /// complex amplitudes.
        /// </summary>
        /// <param name="caReference"> The reference field. Must be globally polarized
        /// and have the same Jones' vector as the field for analysis. </param>
        /// <param name="caAnalysis"> The field for analysis. Must be globally polarized
        /// and have the same Jones' vector as caReference. </param>
        /// <param name="refractiveIndex"> The refractive index of the embedding
        /// medium. </param>
        /// <returns> The relative standard deviation between the two fields. </returns>
        private double ComputeDeviation(
            ComplexAmplitude caReference,
            ComplexAmplitude caAnalysis,
            Complex refractiveIndex)
        {
            // Declare output and intermediate functions:
            double sumOfSquareOfDifference = double.PositiveInfinity;
            double standardDeviation = double.PositiveInfinity;
```

Programming a Module that Computes the Standard Deviation

C# Module: Support Function (Part II)

```
            try
            {
                // Compute the pointwise difference between the two fields:
                ComplexAmplitude caDifference =
                    ComplexAmplitude.SubtractWithResampling(
                    caAnalysis,
                    caReference,
                    InterpolationMethod.SincFFT,
                    InterpolationMethod.SincFFT);

                // Compute the sum of the square of the difference (first
                // step towards calculating the discrete equivalent of the
                // integral):
                sumOfSquareOfDifference = ComplexFieldEvaluation.GetSum(
                    caDifference.Field,
                    ComplexPart.ExtractSquaredAmplitude,
                    false).Abs();
```

Programming a Module that Computes the Standard Deviation

C# Module: Support Function (Part III)

```
// ... and scale it by multiplying by the sampling distance,
// with allowances for one-dimensional fields (yields
// the discrete equivalent of the integral):
if (caDifference.SamplingPoints.X > 1)
{
    sumOfSquareOfDifference *= caDifference.SamplingDistance.X;
}
if (caDifference.SamplingPoints.Y > 1)
{
    sumOfSquareOfDifference *= caDifference.SamplingDistance.Y;
}

// Next, separately, compute the square of the norm of the
// reference field (ditto):
double normOfReferenceField = ComplexFieldEvaluation.GetSum(
    caReference.Field,
    ComplexPart.ExtractSquaredAmplitude,
    false).Abs();
```

Programming a Module that Computes the Standard Deviation

C# Module: Support Function (Part IV)

```
            // ... and, again, scale by multiplying by the sampling distance
            // (ditto):
            if (caReference.SamplingPoints.X > 1)
            {
                normOfReferenceField *= caReference.SamplingDistance.X;
            }
            if (caReference.SamplingPoints.Y > 1)
            {
                normOfReferenceField *= caReference.SamplingDistance.Y;
            }

            // Finally, the relative deviation is given by the ratio of the
            // integral of the square of the norm of the pointwise difference
            // between the two fields, and the integral of the square of the
            // norm of the reference field:
            standardDeviation = sumOfSquareOfDifference / normOfReferenceField;
        }
```

Programming a Module that Computes the Standard Deviation

C# Module: Support Function (Part V)

```
            catch
            {
                ;
            }

            // Return result:
            return standardDeviation;
        }
    }
}
```

Document Information

title	Programming a Module That Computes the Standard Deviation between Two Harmonic Fields
document code	CZT.0009
version	1.0
toolbox(es)	Starter Toolbox
VL version used for simulations	7.4.0.49
category	Feature Use Case
further reading	- How to Work with the C# Module and Example (Computing the Deviation Between Two Fields) - Programming a Module That Smooths the Edges of a Structure